CS522 – Final Exam

take home, 48 hours

You need only 100 (out of 120) points for maximum score

Problem 1 (40 points). Consider the common "letrec" language construct, whose exact syntax here is "letrec $f(x) = e$ in e'", where f and x range over names and e and e' range over expressions. Some functional languages allow more general variants of "letrec", for example where a name rather than a function can be bound or where multiple bindings are allowed; we do not consider those here. This exercise is intended to test your understanding of almost all the material covered in this class, except for the program verification and concurrency parts, so you are not allowed to "compile" the "letrec" construct away into μ or Y and then just reduce its semantics to semantics of language features already covered in the class (though you may want to do that for yourself, as a double check).

8 points Give "letrec" a semantics in the context of untyped λ-calculus, by showing how you can translate it into a corresponding λ-expression and show a β-reduction of factorial of 3 defined using "letrec".

8 points Similarly for combinatorial logic.

6 points Extend simply-typed λ-calculus with a typed variant of the "letrec" above (invent your own syntax) and give its typing rules, its typing algorithm, and its set-of-well-formed-terms closure.

6 points In the context of the item above, give the "typed letrec" an equational semantics, a transitional semantics (aka small-step SOS) and a natural semantics (aka big-step SOS).

6 points Give the "typed letrec" a denotational (aka fixed-point) semantics.

6 points Assuming everything in the lecture notes on parametric polymorphism, use your "typed letrec" above to define a polymorphic fold function. As a reference, here is an SML definition of fold:

```
let rec fold f z = function [] -> z | (x::xs) -> (fold f (f z x) xs);;
```

Problem 2 (40 points). This exercise covers program verification. Consider the program:

```
while (y >= 1) (
  while (even?(y)) (
    x = x * x ;
    y = y / 2) ;
  z = z * x ;
  y = y - 1)
```

`even?(y)` tests whether `y` is an even number.

20 Points Explain what this program does by providing pre- and post-conditions in a Hoare logic style. Then prove the program correct using Hoare logic;

20 Points Explain what this program does by providing pre- and post-conditions in a matching logic style. Then prove the program correct using matching logic. For more on matching logic see `\http://fsl.cs.uiuc.edu/index.php/Matching_Logic`.

Your proofs need not be overly detailed, but enough detail should be included to be convincing that you understand Hoare logic and matching logic, respectively.

Problem 3 (40 points). This exercise tests your understanding of concurrency and its implications. Consider the K-CHALLENGE (or UGLY) language discussed in class.

20 points While the "receive" constructs assumed a variant in which the sender was known but also a variant in which the sender was not known, the "send" constructs always assumed that the recipient was known to the sender. Define variants of "send" where the sent message is not directed to a particular recipient. Feel free to change the syntax of the existing constructs for clarity and also the structure of messages if needed.

20 points Propose a type system for K-CHALLENGE. The current untyped version of this language is quite permissive; in particular, one can send local locations via messages, which the receiving agent may use to access its own store, thus leading to potentially unexpected behaviors. You can make simplifying assumptions about non-concurrent language features; for example: there are two basic types, *int* and *bool*; only output *int*s; λ-expressions are simply typed (so no type inference); drop callcc. With regards to the concurrent features, assume that thread synchronization is done only with *int* values and that messages do not "leak" locations (directly or indirectly via environments).